Managing Virtual Money for Satisfaction and Scale Up in P2P Systems

Jorge-Arnulfo Quiane-Ruiz, Philippe Lamarre, Sylvie Cazalens, Patrick Valduriez

Managing Virtual Money for Satisfaction and Scale Up in P2P Systems*

Jorge-Arnulfo Quiane-Ruiz, Philippe Lamarre, Sylvie Cazalens, and Patrick Valduriez

Atlas group, INRIA and LINA, Universite de Nantes
{quiane, lamarre, cazalens}@univ-nantes.fr, Patrick.Valduriez@inria.fr

ABSTRACT

In peer-to-peer data management systems query allocation is a critical issue for the good operation of the system. This task is challenging because participants may prefer to perform some queries than others. Microeconomic mechanisms aim at dealing with this, but, to the best of our knowledge, none of them has ever proposed experimental validations that, beyond query load or response time, use measures that are outside the microeconomic scope. The contribution of this paper is twofold. We present a virtual money-based query allocation process that is suitable for large-scale super peer systems. We compare a non microeconomic mediation with micro-economic ones from a satisfaction point of view. The experimental results show that the providers' invoice phase is as much important as the providers' selection phase for a virtual money-based mediation.

1. INTRODUCTION

We consider a peer-to-peer (P2P) data management system with mediators that enable consumers to access distributed information providers through queries [6], such as super peer systems. Consumers and providers (for clarity, we refer to both together as participants) are autonomous in the sense that they are free to enter and leave the system at will and do not depend on anyone to do so. The main function of the mediator is to allocate each incoming query to the providers that can answer it. In this context, it is crucial to ensure good system performance, e.g. high throughput or low response times. Nevertheless, participants may get satisfied by the mediator because of certain expectations that are not only performance-related. These participants' expectations are clearly illustrated by Google AdWords [1], which proposes relevant commercial providers to consumers and relevant consumers to commercial providers according to some keywords of their interest.

*Work partially funded by ARA "Massive Data" of the French ministry of research (Respire project) and the European Strep Grid4All project.

To capture this intuition, we have proposed a definition of satisfaction [9] of the participants. The satisfaction of a provider after several query allocations is evaluated considering its intentions w.r.t. the queries that it has been allocated. Satisfaction is defined as a long run notion, to capture the fact that the provider is "globally" satisfied even if it is sometimes allocated queries it does not intend to treat. We have proposed a same kind of definition to characterize the consumers' satisfaction. We have also proposed a mediation process, called $SQLB$ mediation, which allocates a query considering the participants' intentions with respect to the query and their current satisfaction. We implemented the $SQLB$ mediation within a mediator and run experimentations with a single mediator and many participants. The results show that $SQLB$ has very good system performance in maintaining a good satisfaction for participants (as far as they are adequate to the system) and low response times.

However, directly considering the participants' intentions and current satisfaction in the mediation process may have some drawbacks. First, the process suits only in cases where the participants agree to show their intentions with respect to the queries. Second, scaling up to several mediators requires a high message traffic. Indeed, the mediator can no longer calculate the providers' satisfactions itself, because, in case of a provider using several mediators, its satisfaction results from the queries obtained with the different mediators. Thus either each provider has to inform each mediator of its current satisfaction or the mediators have to exchange information about the providers' satisfaction. In both cases, network traffic increases significantly.

A way to regulate query allocation with less information exchange is to use *virtual money*. This latter expression means that this money is totally disconnected from the real money we use in current life. When using virtual money, the providers no longer show their intentions directly, but rather bid on the incoming queries. Thus, the mediation process considers the providers' bids to allocate the queries. In case of several mediators, the same virtual money can be used by the different mediators. It can be considered as the common denominator of the different mediation processes, which use the bids, and possibly other elements to decide the allocation. On the other side, a participant's satisfaction still depends on the work of the different mediators and thus it is still important to satisfy participants.

In this context, several works use microeconomic mechanisms to allocate resources in distributed systems. However, to the best of our knowledge, none of them has ever proposed experimental validations, which, beyond query load

.

or response time, use measures that are outside the microeconomic scope. Thus, the main contribution of this paper is twofold. First, we present a virtual money-based query allocation process that *(i)* is participant-centric, *(ii)* behaves as good as $SQLB$, and *(iii)* scales up to several mediators. Second, from a methodological point of view, it is interesting to compare a non microeconomic mediation (namely $SQLB$) with microeconomic ones (namely $FPSB$, VM_bQA, and VM_bQA^+), using satisfaction as a "money independent" evaluation measure in both cases.

This paper is organized as follows. In Section 2, we give some preliminary concepts and definitions. In Section 3, we discuss the use of virtual money when designing a mediation process. We present in Section 4 a virtual money-based query allocation process and improve this mediation process in Section 5. In Section 6, we discuss how the mediation process we propose can normally perform in super peer systems. Finally, we present related work and conclude this paper in Sections 7 and 8, respectively.

2. BACKGROUND

In this section, we make precise the query allocation problem in environments where participants are autonomous. With this in mind, in Section 2.1, we discuss in more detail the system that we consider and describe, in Section 2.2, how participants compute their *satisfaction*. Finally, in Section 2.3, we state the query allocation problem in such systems and, in Section 2.4, we present a solution to this problem, which we use as baseline mediation.

2.1 System Model

The distributed system we wish to control consists of a mediator m and of a set $\mathcal{I}$ of autonomous participants, who may play two different roles: consumer and provider. Participants are autonomous in the sense that they may enter and leave the system at any time and are free to join and quit a mediator $m \in M$ at will. A mediator m allocates queries of a set C of consumers (with $C \subseteq \mathcal{I}$) to a set P of providers (with $P \subseteq \mathcal{I}$) and returns results or a set of selected providers (depending on the system architecture) to consumers. As a participant may play both consumer's and provider's roles, it is possible to have $C \cap P \neq \emptyset$.

A consumer poses a query to the mediator when it cannot locally perform the query or just because it has certain gains by outsourcing such a query. For example, a consumer may query the system to perform a given application because *(i)* it has not enough resources to run the application, or *(ii)* other participant (a provider) performs the application faster. Queries are formulated in a format abstracted as a triple $q = < c, d, n >$ such that $q.c \in C$ is the identifier of the query initiator (the consumer), $q.d$ is the description of the task to be done, e.g. a SQL statement, and $q.n \in \mathbb{N}^*$ is the number of providers to which consumer $q.c$ wishes to allocate its query.

Providers in P have different processing capabilities and capacities for performing incoming queries. On the one hand, different processing capabilities mean that providers provide different data and thus produce different results for a same query. On the other hand, different processing capacities mean that some providers are more powerful than others and thus can treat more queries per time unit. The *capacity* of a provider $p \in P$, $cap_p > 0$, denotes the number of computational units (expressed e.g. in time units) that it has to perform queries. In the same way, a query q has a *cost*, $cost_p(q) > 0$, that represents the computational units that it consumes at provider p. Thus, the *utilization* of a provider $p \in P$ at a given time t, $\mathcal{U}_t(p)$, is defined as p's load with respect to its capacity. In other words, function $\mathcal{U}_t(p)$ denotes the total cost of the queries that have been allocated to p and have not already been treated at time t.

A participant $i \in \mathcal{I}$ is free to express its *intention* to allocate and perform a query and it is up to it to compute its own *intentions* by combining different local and external criteria (e.g. load, preferences, and reputation). The *intentions* are denoted by values in the interval of $[-1..1]$. If an *intention* is positive, the greater the value, the greater the desire of a participant that a possibility becomes a reality. In contrast, if an *intention* is negative, the closer the value from -1, the greater the desire of a participant to do not see a possibility becomes a reality. It is worth noting that even if participants can express their *intentions* it does not mean that they can refuse queries.

2.2 Participant's Satisfaction

Intuitively, a participant $i \in \mathcal{I}$ is satisfied with the query allocation process if the latter meets its *intentions* in the long-run. The *satisfaction* of a participant may have a deep impact on the system, because the participant may decide whether to stay or to leave the system based on it. Recently, we proposed a complete model that characterizes autonomous participants by formally defining the adequation, satisfaction, and allocation satisfaction of a participant [9]. We briefly present in this section the intuitions of a participant's satisfaction. To discuss further this notion and present the adequation and allocation satisfaction notions is beyond the scope of this paper.

The consumer's satisfaction allows to evaluate whether the mediator is allocating the queries of a consumer to the providers that the consumer wants to deal with. To compute its satisfaction, $\delta_s(c)$, a consumer $c \in C$ proceeds as follows,

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \frac{1}{n} \Big(\sum_{p \in \widehat{P_q}} \big(\overrightarrow{CI}_q[p] \ + 1\big) \Big/ 2 \Big) \qquad (1)$$

where n stands for $q.n$, IQ_c^k denotes the set of k last queries issued by c, $\widehat{P_q}$ is the set of providers that performed q, and $\overrightarrow{CI}_q[p]$ is the intention expressed by c towards provider p.

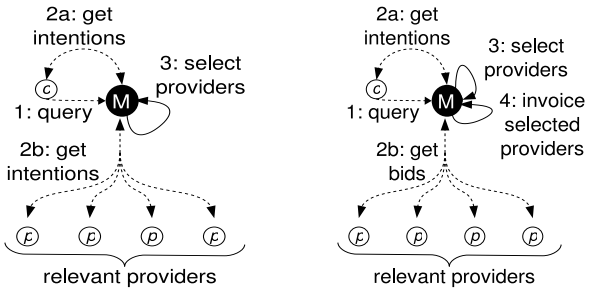
The provider's satisfaction evaluates whether the mediator is giving queries to a provider according to its expectations (those of the provider) so that it fulfills its objectives. Thus, as for consumers, a provider is simply not satisfied when it does not get what it expects. To evaluate so, a provider $p \in P$ computes its satisfaction, $\delta_s(p)$, as follows,

$$\delta_s(p) = \begin{cases} \Big(\big(\frac{1}{||SQ_p^k||} \sum_{q \in SQ_p^k} \overrightarrow{PPI}_p[q] \ \big) + 1 \Big) \Big/ 2 \\ 0 \qquad\qquad\qquad\qquad if\ SQ_p^k = \emptyset \end{cases} \qquad (2)$$

where SQ_p^k denotes the set of queries performed by p and $\overrightarrow{PPI}_p[q]$ is the intention expressed by p towards query q. Notice that both above equations may be stated with respect to the participants' preferences as well.

2.3 Query Allocation Problem

Let P_q denote the set of providers registered to mediator m, which does not appear in the notation for simplicity, and



(a) SQLB-based (b) virtual money-based

Figure 1: System architecture. Peer M denotes the mediator, c a consumer, and p a provider.

that can deal with an incoming query q. We only consider the arrival of those queries for which $P_q \neq \emptyset$. Thus, the allocation of query q is denoted by a vector $All\overrightarrow{oc}_q$ of length N (with $N = ||P_q||$) such that,

$$\forall p \in P_q,\ All\overrightarrow{oc}[p] = \begin{cases} 1 & \text{if } p \text{ gets the request} \\ 0 & \text{otherwise} \end{cases}$$

Set $\widehat{P_q}$ denotes the set of providers such that $All\overrightarrow{oc}_q = 1$, whose length is $\min(q.n, N)$. In other words, if the number of providers that are able to deal with a query q is smaller or equal than the number of desired providers by a consumer, all the providers in P_q must perform q. Given all the above, we can then state the query allocation problem in mono-mediator systems with autonomous participants as follows.

Definition 1. **Query Allocation Problem** *Given a mediator m confronted to $||\mathcal{I}||$ autonomous participants, m should allocate each incoming query q to a set $\widehat{P_q}$ such that $||\widehat{P_q}|| = min(q.n, N)$ and good system performance as well as participants' intentions are satisfied in the long-run.*

2.4 SQLB Mediation

SQLB allocates queries by balancing consumers' and providers' *intentions* in accordance to their *satisfaction*. In that work, we experimentally proved that *SQLB* ensures good system performances and satisfies, in the long-run, the participants' *intentions*, which allows it to scale up in environments with autonomous participants. This is why we consider it as a baseline method in the remainder of this paper. The general system architecture is shown by Figure 1(a) and the principle of *SQLB* is the following. A consumer (the user) sends a query q to the mediator, which must find set P_q and ask for $q.c$'s intentions regarding each provider in P_q. Also, it must ask for P_q's *intention* for performing q. Once the mediator obtains such information, it computes the score of each provider $p \in P_q$ by making a balance between $q.c$'s and p's *intentions* and computes the ranking of providers in P_q based on their score. Finally, it allocates q to the $q.n$ best scored providers in P_q and informs all participants of the mediation result.

3. USE OF VIRTUAL MONEY

We now consider the introduction of *virtual money* within the system so as to allow a system to scale up more easily. Notice that this money is purely virtual. This point has to be stressed upon for two main reasons. First, we do not focus on any particular business model: we only use the virtual money as a means to regulate the query allocation in the system. Indeed, after a consumer has decided which providers it chooses, it might give *real money* to them because it uses their services. This point is far beyond the focus of this paper. Second, when using real money, one can assume that consumers and providers get money from elsewhere. For example, when designing an auction mechanism for e-commerce one can assume that people spend the money they have earned by working (in real life). When dealing with virtual money, one can no longer make such assumptions. Therefore, in our case, to correctly regulate the system, we must precise *the way in which the virtual money circulates* within it. This is a macro-economic concern for which we adopt a simple solution for the mechanism we propose. First, only the providers and the mediator deal with virtual money, *i.e.* the consumers still show their intentions. The policy used to regulate the money flow depends on the mechanism. We decide that, in case the mediator earns money while the providers get poorer and poorer, the mediator distributes the money to the providers; That is, to avoid that the lack of virtual money influences the regulation effect of our mechanism, when the virtual money earned by the mediator is above some threshold, it is distributed again among all providers in an equitable way.

Now, designing the allocation mechanism used by the mediator is a micro-economic concern. Providers no longer express their *intentions* directly to the mediator. Instead, they have to express their intentions through *bids*, which also considers their current virtual money balance as well as their strategy to bid. Notice that the general architecture of the virtual money-based system is almost the same that when one does not use virtual money (see Figure 1(b)). The only thing that changes is that the mediator now asks providers for their *bids* instead of their *intentions* and has to *invoice* providers at the end of each query allocation. Therefore, the design of a specific allocation mechanism first requires to define the way in which the mediator merges the consumers' *intentions* with the providers' *bids* so as to select providers. This is the *selection* phase. Second, we consider in this work systems where providers have to "pay" for performing or receiving queries[1]. Thus, in such models, one has to define the way in which the mediator invoices providers after each query allocation. This is the *invoicing* phase. Finally, one has to define how providers compute their *bids*.

4. VM$_B$QA MEDIATION

We present in this section *Virtual Money-based Query Allocation* (VM_bQA), a particular query allocation method that takes into account both consumers' *intentions* and providers' *bid*. VM_bQA assumes that consumers show their *intentions* (denoted by vector $\overrightarrow{CI}$) to the mediator, while providers bid on queries instead (which is a means to reflect their *intentions* while keeping them private). In Section 4.1, we define the process itself and then define how a provider computes its *bids* in Section 4.2. Finally, we evaluate in Section 4.3 how well VM_bQA allocates queries among providers.

4.1 Definition of the Process

Let us consider the allocation of some query q initiated

[1]This is in some way similar to an invoicing such as the one of *Google AdWords* [1], except we consider virtual money.

by some consumer $c \in C$. The providers in P_q bid on q and such bids are not public to others. The bids are represented by a vector $\overrightarrow{B}$, with $\overrightarrow{B}[p] \in \mathbb{R}$ for all $p \in P_q$. If a bid is positive, the higher it is, the more p wants to be allocated q. If it is negative, the lower it is the less p wants to treat q. Intuitively, the *bid* of p reflects its *intention* to perform q. This should lead to the providers' *satisfaction*. We detail the way in which VM_bQA allocates queries among providers in Section 4.1.1 and define the way in which it invoices providers in Section 4.1.2.

4.1.1 Selection phase

Let us remember that a query q is allocated to the $min(n, N)$ best providers, which are given by vector of ranking $\overrightarrow{R}$. Intuitively, $\overrightarrow{R}[1] = p$ if and only if p is the best ranked, $\overrightarrow{R}[2]$ stands for the second best ranked and so on. Hence, $All\overrightarrow{oc}_q[p] = 1$ if and only if $\exists i, \overrightarrow{R}[i] = p$ and $i \leq min(n, N)$. Traditionally, vector $\overrightarrow{R}$ is computed with respect to the providers' *bids* only. Then, a simple solution to select providers may be the *first-price sealed-bid* approach, which selects the provider having made the highest *bid*. However, this is a provider-centric approach that penalizes consumers' intentions. Thus, to satisfy consumers, VM_bQA allocates queries based on the providers' *level* and computes vector $\overrightarrow{R}$ with regards to these levels (denoted by vector $\overrightarrow{L}$) such that, given an incoming query q, the provider with the highest level is allocated q. We formally define providers' level, denoted by vector $\overrightarrow{L}$, concerning a query q as Definition 2. Given this definition, a query might be allocated to a provider that did not want it. We call this an imposition case, otherwise, we have a competition case.

DEFINITION 2. **Provider's Level** *Given an incoming query q, the level of each $p \in P_q$ is defined as follows,*

$$\overrightarrow{L}[p] = \begin{cases} (\overrightarrow{B}[p] + 1)^{\omega} \times (\overrightarrow{CI}_q[p] + 1)^{1-\omega} & if\ \overrightarrow{B}[p] \geq 0 \\ -(-\overrightarrow{B}[p] + 1)^{\omega} \times (\overrightarrow{CI}_q[p] + 1)^{\omega-1} & otherwise \end{cases}$$

Parameter ω, whose values are in the interval $[0..1]$), ensures the balance between consumers' *intentions* and providers' *bids*. In $SQLB$ a mediator set this parameter according to participant's satisfaction to regulate the system. When using virtual money, this is no more necessary because the virtual money itself is a natural means of regulation. Thus, in VM_bQA, the mediator sets ω according to the importance it wants to give to consumers' *intentions* and providers' *bids*. If $\omega = 0$, only the consumer's *intentions* are considered by the mediator, thus leading to providers dissatisfaction. Conversely, if $\omega = 1$, the mediator only considers bids, leading to consumers dissatisfaction. This is why the mediator should set parameter ω according to the balance between both consumers' and providers' *satisfaction* that it wants to reach.

Table 1 shows the case of a competition. The consumer asks for two providers, and more than two of them bid positively. Providers p_5 and p_3 are allocated the query because they get the two highest levels, respectively 2.28 and 2.24. Notice that the consumer's *intention* with respect to p_5 is lower than its intention with respect to p_3. Thus, p_5 only got the query because of its bid (2.25) which is higher than p_3's bid (1.79), meaning that it wanted the query more than p_3. Table 2 shows an imposition case where no provider but

Table 1: VM_bQA: a competition case

		p1	p2	p3	p4	p5	m
init	CI	0,9	0,8	0,8	0,6	0,6	
	bal	16,05	15,89	17,89	15,51	22,51	0,00
q5	B	1,60	1,59	1,79	1,55	2,25	
	Level	2,22	2,16	2,24	2,02	2,28	
	Rank	3	4	2	5	1	
	Alloc			*		*	
	Trans			**1,75**		**2,09**	
	bal	16,05	15,89	**16,14**	15,51	**20,42**	**3,84**

$\omega = 0.5$; $n = 2$

Table 2: VM_bQA: an imposition case

		p1	p2	p3	p4	p5	m
init	CI	0,9	0,8	0,8	0,6	0,6	
	bal	18,16	18,00	20,00	17,63	17,63	0,00
q4	B	-18,00	-12,00	-8,00	0,35	-2,00	
	Level	-3,16	-2,69	-2,24	1,47	-1,37	
	Rank	5	4	3	1	2	
	Alloc				*	*	
	Trans	**2,11**	**2,11**	**2,11**	**2,11**	**-4,89**	
	bal	**16,05**	**15,89**	**17,89**	**15,51**	**22,51**	**3,56**

$\omega = 0.5$; $n = 2$

p_4 wants to treat the query, whereas the consumer asks for two providers. Provider p_5 is imposed the query because of both its bid (which is the highest negative bid) and the consumer's *intention* with respect to it, which leads to the value 1.47 of its level.

4.1.2 Invoicing phase

As start point, a natural strategy to invoice a provider p that has been allocated a query q is that p pays what it has bid for q, which is denoted by $\overrightarrow{B}[p]$. Thus, given a query q, the mediator computes the providers' invoice as follows.

DEFINITION 3. **Provider's Invoice** *If a provider $p \in P_q$ is allocated a given query q, then it must pay the amount of virtual money it has offered to get q. Then, after the payment for q, the new virtual money balance of p, bal_p, is set as follows, $bal_p = bal_p - \overrightarrow{B}[p]$*

4.2 Bid Computation

A simple way to obtain providers' *bid*, is that each provider maintains a local bulletin board, which contains a billing rate for its resources based on its *preferences* to perform queries (denoted by function prf). Then, a provider's *bid* for getting a query may be the product of its current utilization by the billing rate, i.e. its *preferences* for the query (such as in [13]). In our case, the context of a provider is more complex: we have to consider its *preferences*, load, current *satisfaction*, and current virtual money balance. Thus, given a query q, a provider first works out its *intention* to perform q by considering its *preferences* (denoted by function $prf_p(q)$, whose values are in the interval $[-1..1]$), its current *utilization*, and its current *satisfaction*. Intuitively, on the one hand, a provider may not pay so much attention to its preferences and accepts sometimes queries it does not desire if it is satisfied. On the other hand, if a provider is not satisfied, it may focus on its *preferences* to obtain desired queries. Then, a provider computes its *intention* as follows.

DEFINITION 4. **Provider's Intention** *Given an incoming query q, a provider $p \in P_q$ computes its intention towards*

q, $pi_p(q)$, *as follows,*

$$pi_p(q) = \begin{cases} (prf_p(q)^{1-\delta_s(p)}) \times (1-\mathcal{U}_p(t))^{\delta_s(p)}, & if(prf_p(q) > 0) \wedge \\ & \wedge (\mathcal{U}_p(t) < 1) \\ -\big((1-prf_p(q)+1)^{1-\delta_s(p)} \times & \\ \times(\mathcal{U}_p(t)+1)^{\delta_s(p)}\big) & else \end{cases}$$

Once a provider obtains its *intention* with respect to a query, it then proceeds to work out its *bid* to perform such a query. Intuitively, the bid of a provider p is the product of its *intention* by its current virtual money balance (bal_p). Nonetheless, such a simple procedure may lead a provider to spend all, or almost all, its money on only one query. To avoid such a behavior, a provider offers at most a defined percent of its current virtual money balance, denoted by constant c_0, $0 < c_0 \leq 1$. Then, we define a provider's *bid* as in Equation 3, where constant c_1 is set to the initial virtual money balance of a provider.

DEFINITION 5. **Provider's Bid** *Given an incoming query q, a provider $p \in P_q$ computes its bid to perform q, $b_p(q)$, as follows,*

$$b_p(q) = \begin{cases} pi_p(q) \cdot bal_p \cdot c_0 & if\ pi_p(q) > 0 \\ pi_p(q) \cdot c_1 & otherwise \end{cases} \quad (3)$$

The idea behind the above definition is that a provider always sets a positive *bid* when it desires to perform queries and it is not *overutilized*, otherwise it sets a negative *bid*. Notice that in traditional economic-based methods providers just do not bid (or give a null bid) when they do not desire to perform a query. However, this do not allow them to express how much unpleasant it is for them to perform a query and how much overloaded they are. Thus, Equation 3 allows a provider to preserve its *preferences* while good response times are also ensured to consumers. At first glance, there is no difference between providers' bids and intentions, but by showing bids providers can keep private its real intentions.

4.3 Evaluation of VM_bQA

Our aim in this evaluation is to analyze the impact of using virtual money to address the query allocation problem defined in Section 2.3. To clearly see such an impact, we compare VM_bQA mediation with $SQLB$ mediation, which has been proved to perform well in autonomous environments. Also, to analyze the possible loss that VM_bQA may have with respect to providers due to the attention it pays to consumers' intentions, we compare VM_bQA with a provider-centric mediation: the *first-price sealed-bid* mediation ($FPSB$ for short). Unlike VM_bQA that allocates queries by balancing consumers' intentions with providers' bid, $FPSB$ allocates each incoming query to those providers that have made the highest bid. In contrast, as in VM_bQA as in $FPSB$, a provider pays for a query what it has bid. In Section 4.3.1, we describe the experimental setup and, in Section 4.3.2, present the experimental results.

4.3.1 Evaluation Setup

The system consists of 200 consumers, 400 providers, and only one mediator allocating all the incoming queries. We assigned sufficient resources to the mediator so that it does not cause bottlenecks in the system. We assume that consumers compute their intentions by considering only their preferences. Concerning a provider in $SQLB$ mediation, on the one hand, we assume that it computes its intentions as defined in [9]. On the other hand, a provider in both $FPSB$ and VM_bQA mediations computes its *bid* as defined in the Section 4.2. To study all three mediations, we implemented an observer that computes participants' satisfaction as presented in Section 2.2. It initializes participants' satisfaction with a value of 0.5, which evolves with their last 200 issued queries and 500 queries that have passed through providers (i.e. $k = 200$ for a consumer and $k = 500$ for a provider).

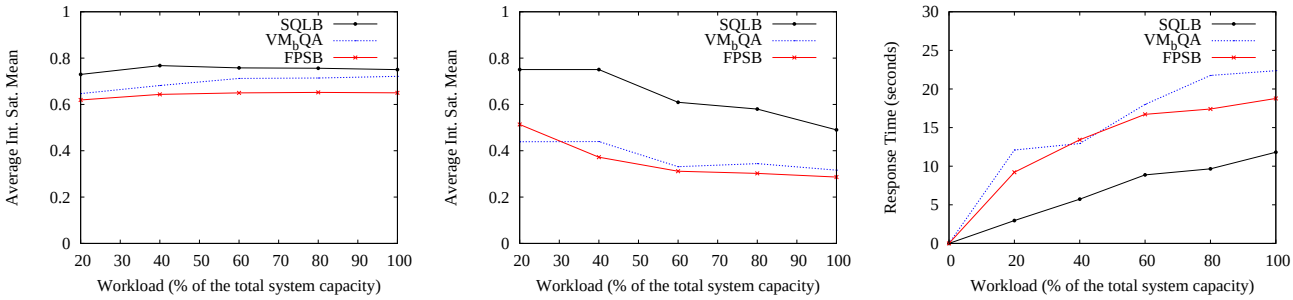
We set the heterogeneity of the providers' capacity in accordance to the results presented in [11]. We generate around 10% of providers with *low*-capacity, 60% with *medium*, and 30% with *high*. The *high*-capacity providers are 3 times more powerful than *medium*-capacity and still 7 times more powerful than *low*-capacity providers. We generate two classes of queries that *high*-capacity providers perform in 1.3 and 1.5 seconds, respectively. We consider that queries arrive to the system in a *Poisson* distribution, as found in dynamic autonomous environments [5]. To simulate high heterogeneity of the consumers' *preferences* for allocating their queries to providers, we divide the set of providers into three classes according to the interest of consumers: to those that consumers have *high* interest (60% of providers), *medium* interest (30% of providers), and *low* interest (10% of providers). Consumers randomly obtain their *preferences* between .34 and 1 for *high*-interest providers, between $-.54$ and .34 for *medium*-interest providers, and between -1 and $-.54$ for *low*-interest providers. On the other side, to simulate high heterogeneity of the providers' *preferences* towards the incoming queries, we also create three classes of providers: those that have *high* adaptation (35% of providers), *medium* adaptation (60% of providers), and *low* adaptation(5% of providers). Here, adaptation stands for a provider's degree of interest to perform incoming queries. Providers randomly obtain their *preferences* between $-.2$ and 1 (*high*-adaptation), between $-.6$ and .6 (*medium*-adaptation) or between -1 and .2 (*low*-adaptation).

Concerning participants' departures, we assume on the one hand that a consumer leaves the system, by *dissatisfaction* if its *satisfaction* is smaller than .4. On the other hand, we assume a provider may leave the system by three reasons: *(i) dissatisfaction* if its *satisfaction* is smaller than 0.3, *(ii)* by query *starvation* if, in an interval of 1 minute, it does not perform a set of queries towards which it has a preference of at least 0.2 in average, and *(iii)* by *overutilization* if its *utilization* is greater than 1.5 if it is a high-capacity provider, 2.5 if it is a medium-capacity provider, or 3 if it is a low-capacity provider. Without any loss of generality, the participants' expectations, in the long run, are static in our simulations. We assume this to evaluate all three mediations in a long-term trend.

Finally, let us make the following assumptions. First, as our main focus is to study if VM_bQA performs equal or better than $SQLB$, we do not consider the bandwidth problem and assume that all participants have the same network capacities. Second, for the sake of simplicity, we assume that consumers only ask for one informational answer (i.e. $n = 1$) and that all the providers in the system are able to perform all the incoming queries. Above assumptions do not harm with the generality of our proposal.

4.3.2 Evaluation Results

Before going on to discuss the results, let us say that, for space reasons, we do not present all results we obtained from



(a) Consumers' satisfaction. (b) Providers' satisfaction. (c) Response times.

Figure 2: *(a-b)* **participants'** *satisfaction* **results and** *(c)* **ensured response times, for different workloads.**

this experimentation but the most relevant. That is, we only discuss the results concerning the participants' satisfaction and response times ensured by all three mediations.

Figure 2(a) illustrates the consumers' satisfaction results. We can observe that VM_bQA satisfies consumers much better than $FPSB$ and is not so far from $SQLB$. Obviously, VM_bQA outperforms $FPSB$ because $FPSB$ has no consideration of the consumers' intentions. However, when regarding the providers' satisfaction in Figure 2(b), we can observe that VM_bQA once again outperforms $FPSB$. This proves that VM_bQA has no loss in performance w.r.t. providers' satisfaction while it has benefits from the consumers' point of view. Nonetheless, during our experimentations we observed that $FPSB$ better balances queries among providers than VM_bQA. This is because VM_bQA also considers the consumers' intentions and thus, given the experimental setup, VM_bQA tends to allocate queries to the most reputed providers (which are those providers that consumers prefer in general) causing a query load imbalance in the system. This is reflected in the response time ensured by both two mediations, which is illustrated by Figure 2(c). We can observe in those results that $FPSB$ ensures lower response times than VM_bQA. It is however so far from $SQLB$, which outperforms it by a factor of 2.

5. IMPROVEMENT OF VM$_B$QA

Although VM_bQA has no loss in performance concerning providers' satisfaction and only a small loss concerning response times w.r.t. $FPSB$, it is still far from $SQLB$ in all cases: participants' satisfaction and ensured response times. An intuition to improve VM_bQA is to change the way in which it invoices providers. In this section, we present another way to invoice providers [4] in Section 5.1 and evaluate it in Section 5.2.

5.1 Invoicing Providers

We now discuss another manner to compare bids to rank providers regarding their bids. Notice that bids cannot be directly compared because of consumers' *intentions*. Thus, to overcome this difficulty, the *theoretical bid* is introduced ($B^{Th}(p,l)$), which corresponds to the amount that provider p should bid for reaching level l. With $\omega \neq 0$ and $\alpha = 1$ if $l \geq 0$, and $\alpha = -1$ otherwise, $B^{Th}(p,l)$ is given by the following formula,

$$B^{Th}(p,l) = \alpha max(((\alpha \times l)^{\frac{1}{\omega}}(\overrightarrow{CI}_q[p]+1)^{\frac{\alpha(\omega-1)}{\omega}} - 1), 0) \quad (4)$$

For example, in Table 1, we have already noticed that provider p_5 gets a level slightly higher than p_3's, because of its higher *bid* and despite the lower consumer's *intention*. In fact, to come exactly to p_3's level, p_5 should bid 2.136 (theoretical *bid*). Now, given a provider $p' \in P_q$, $\overrightarrow{PTr}[p,p']$ denotes what p' owes due to the allocation of q to $p \in P_q$ ($\overrightarrow{PTr}[p,p'] = 0$ if p is not allocated q). Then, the total amount paid by p' is defined by a sum (Equation 5).

$$\forall p' \in P_q, \quad \overrightarrow{Trans}[p'] = \sum_{p \in P_q} \overrightarrow{PTr}[p,p'] \quad (5)$$

There is a competition among providers when there are enough providers that want to be allocated the query. In that case, each of them pays the amount of its theoretical bid to reach the level of the best provider which has not been selected (in the spirit of a generalized Vickrey auction [14] except that the consumer's *intention* is considered). In Table 1, only providers p_5 and p_3 pay (respectively 2.09 and 1.75) to the mediator, thus decreasing their own money balance (*bal*). A requisition case occurs when at least one provider is imposed the query. Obviously, being imposed does not meet at all the expectations of the imposed providers. Hence, to keep them satisfied in the long run, the idea is then to distribute the cost of the imposition on *all* the providers in P_q (in the spirit of [12], but also considering the consumer's *intentions*). Then, having obtained a reward, the imposed providers are more likely, in the future, to obtain the queries they expect (because they have more money) so leading to their satisfaction. The formal definitions of the transfers in the imposition case are as in definition below.

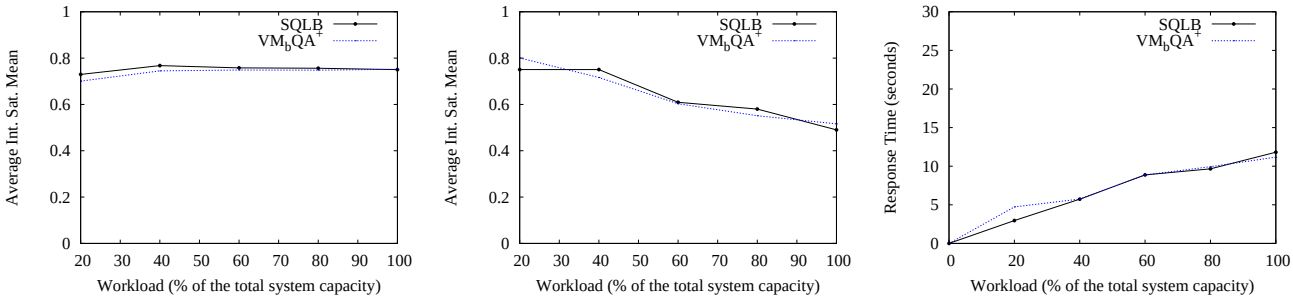
Definition 6. **Partial Transfers in a Requisition Case** *If provider $p \in P_q$ is allocated q and $\overrightarrow{B}[p] < 0$, then for all $p' \in P_q$:*

$$\overrightarrow{PTr}[p,p'] = \begin{cases} \dfrac{-B^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+2,N)]])}{N} & if p \neq p' \\ B^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+1,N)]]) & else \\ \quad - \dfrac{B^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+2,N)]])}{N} & \end{cases}$$

In the example of Table 2, p_5 is imposed and thus gets a 4.89 reward. All the providers contribute to this reward. Notice also in both Table 1 and Table 2 that the mediator gets some money left, which is of no use for it. We discussed this point in Section 3.

5.2 Evaluation of VM$_b$QA$^+$

We implemented VM_bQA^+ mediation, which selects providers as VM_bQA does but invoices them as defined in



(a) Consumers' satisfaction. (b) Providers' satisfaction. (c) Response times.

Figure 3: *(a-b)* **participants'** *satisfaction* **results and** *(c)* **ensured response times, for different workloads.**

previous section. We compare VM_bQA^+ with $SQLB$ to evaluate whether it can ensure the same system's performance as $SQLB$. Particularly, we evaluate whether the way in which VM_bQA^+ invoices providers improves the performance of VM_bQA. To be comparable with evaluated mediations in Section 4.3, we consider the same experimental setup we used in Section 4.3.1 to run these experimentations.

5.2.1 Results

We can observe in Figures 3(a) and 3(b) that VM_bQA^+ ensures the same participants' satisfaction, with only a quite small difference for low workloads, as $SQLB$. This is because the way in which VM_bQA^+ invoices providers avoids that some providers monopolize queries, thereby allowing to better balance queries among providers. This allows VM_bQA^+ better satisfying providers without penalizing consumers' intentions. Figure 3(c) illustrates the response times ensured by VM_bQA^+. We observe that VM_bQA^+ has the same performance as $SQLB$, however we can observe that for a workload of 20% of the total system capacity $SQLB$ slightly outperforms VM_bQA^+. Even this small difference we can see that VM_bQA^+ generally performs as well as $SQLB$. Thus, we can conclude that we can introduce virtual money, without any loss of system's performance, to regulate a system as long as we care about the way in which providers are selected and invoiced.

6. DEALING WITH SEVERAL MEDIATORS: SUPER-PEER NETWORKS

We experimentally proved in previous section that VM_bQA^+ performs as well as $SQLB$ in mono-mediator systems. In this section, we demonstrate that VM_bQA^+ performs well in multi-mediator systems, such as in super-peer systems. A super-peer network is a pure peer-to-peer (P2P) network, i.e. an unstructured P2P network, but each peer in the network is actually a super-peer. A super-peer operates as a server to a cluster of peers (participants) and as equal participant with respect to other super-peers. We call a set of participants with at least one of them playing the role of super-peer a virtual organization (VO). Figure 4 illustrates a super-peer network with 6 VOs. Notice that, one can see a VO as a mono-mediator system. Thus, one can directly use $SQLB$ to perform query allocation inside a VO so as to satisfy participants while ensuring good system performance. Nevertheless, a super-peer becomes a single point of failure for its VO as well as a potential performance and scalability bottleneck. To avoid this, one can introduce redundancy into the assignment of super-peers, which has been experimentally proved to have gains in performance [15]. A $x-redundant$ VO is defined as follows.

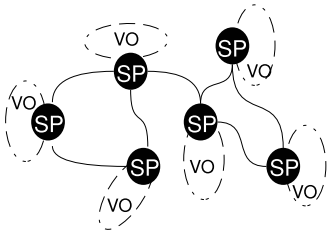


Figure 4: Super-peer network with 6 super-peers, which are represented by the SP peers.

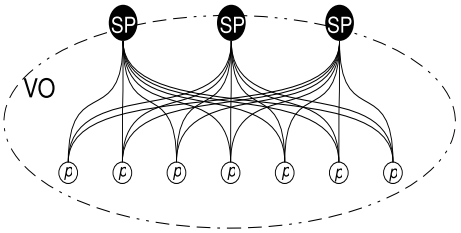


Figure 5: A virtual organization (VO) with 3-redundancy, where p peers represent those participants that do not play the super-peer role.

DEFINITION 7. **x-redundant VO** *Let $\mathcal{I}'$ denote the set of participants in a VO and $\mathcal{I}'_{sp}$ denote the set of participants connected to super-peer sp. A VO is then said to be $x-$ redundant if and only if there is a set $\mathcal{X}$ of x participants, with $\mathcal{X} \subseteq \mathcal{I}'$ and $x > 1$, playing the role of super-peer and each participant in $\mathcal{I}' \backslash \mathcal{X}$ is connected to each super-peer sp in $\mathcal{X}$, such that $\forall sp \in \mathcal{X}$, $\mathcal{I}'_{sp} = \mathcal{I}'$.*

Figure 5 illustrates a $x-redundant$ VO with 3 super-peers, which form a single super-peer for the VO. In this case, when a VO is $x-rendundant$, $SQLB$ cannot perform as well as in a mono-mediator system because the participants' satisfaction computed by each super-peer in $\mathcal{X}$ is local and hence different. A simple solution may be that the x super-peers in a VO frequently exchange messages to update participants' satisfaction. However, because of network latency, even if mediators continually update participants' satisfaction, there will be always a time period where

participants' satisfaction is not the same at all mediators. Moreover, these update messages considerably increase the network traffic and hence hurt response times, which does not allow the system to scale up. In contrast, VM_bQA^+ has no message to exchange among mediators because the system regulation is done by the flow of virtual money instead of by satisfaction. In other words, VM_bQA^+ has no additional cost when performing in a multi-mediator system. Therefore, VM_bQA^+ allows a VO, and hence a super-peer system, to scale up with no loss in system performance. We state this in the following Theorem.

THEOREM 1. *VM_bQA^+ always satisfies (i) consumers and (ii) providers in a $x-redundant$ VO as well as in a mono-mediator system.*

PROOF. Consider a $x-redundant$ VO, denoted by S_{vo} and a mono-mediator system, denoted by S_m, consisting of the same set of participants $\mathcal{I}'$. Consider also that the incoming queries in S_{vo} are the same to those arriving in S_m. We prove both (i) and (ii) by contradiction.

(i) Assume to the contrary that, for some query q, consumer $q.c \in \mathcal{I}'$ is not equally satisfied by S_{vo} and S_m. If this is the case, we can know, by Equation 1, that S_{vo} allocated q to a set $\widehat{P_q}'$ such that $\exists p \in \widehat{P_q} : p \notin \widehat{P_q}'$, where $\widehat{P_q}$ is the set of providers selected by S_m. Hence, we can know that the set of relevant providers found by S_{vo} is different to the set found by S_m. This implies that provider p is not connected to the super-peer that allocated q in S_{vo}, which contradicts the definition of a $x-redundant$ VO.

(ii) Assume to the contrary that a provider $p \in \mathcal{I}'$ is not equally satisfied by S_{vo} and S_m. Then, by Equation 2, we can know that p did not perform the same set of queries in S_{vo} as in S_m. This means that p is not connected to all super-peers in S_{vo} so as to receive all queries it can perform, which contradicts the definition of a $x-redundant$ VO. □

To discuss how queries may be forwarded to other VOs is well beyond the scope of this paper.

7. RELATED WORK

Several approaches based on microeconomics have been proposed [2, 3, 8, 13]. A survey of economic models for various aspects of distributed system is presented in [3]. In economic models the notion of *utility* is clearly linked to satisfaction. Nonetheless, it is generally reduced to the allocation of only one query and to monetary concerns. In the field of distributed rational decision making [10], most of the processes are *individually rational*: the utility of any participant in the process is no less than the utility it would have by not participating. This property is not relevant in contexts where some participants may be imposed, which implies having a lower utility in participating. Furthermore, participants may have the interest that the system be effective and, in certain query allocations, some participants may be interested in participating, even if this means to have a low utility. Thus, to consider participants' *satisfaction* is still relevant in these contexts because it is a long-run notion. In [7], the authors focus on the optimization algorithms for *buying* and *selling* query answers, and the negotiation strategy. However, this way of dealing with subqueries optimization is orthogonal to our proposal and one may combine them to improve performance. In [4], we proposed an economic *flexible* mediation that allocates queries by taking into account the *quality* and *bids* of providers, but it is inherently assumed that consumers are only interested in the quality of providers. In a recent work [9], we provide a query allocation framework for distributed information systems with autonomous participants, but we considered VOs with only one super-peer, i.e. mono-mediator systems.

8. CONCLUSION

We addressed the query allocation problem in super-peer data management systems. We presented a virtual money-based query allocation mediation that behaves as good as $SQLB$. We compared a non micro-economic mediation ($SQLB$) with micro-economic ones ($FPSB$, VM_bQA, and VM_bQA^+), by evaluating the participants' satisfaction and response times. From the results point of view, we showed that, with both an adequate selection and invoice, a virtual money based mediation shows results that are as good as those of $SQLB$. This enables scaling up to a super peer network where all the peers use the same virtual money, without any additional cost for the allocation of a query.

9. REFERENCES

[1] Google adwords, http://adwords.google.com.

[2] R. K. Dash, P. Vytelingum, A. Rogers, E. David, and N. R. Jennings. Market-Based Task Allocation Mechanisms for Limited Capacity Suppliers. *IEEE Transactions on Systems*, 37(3):391–405, 2007.

[3] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems. In S. H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[4] P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. In *Proceedings of the Cooperative Information Systems Confeference (CoopIS)*, 2004.

[5] E. P. Markatos. Tracing a large-scale peer to peer system: An hour in the life of gnutella. In *Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[6] T. Özsu and P. Valduriez. *Principles of Distributed Database Systems, Second Edition*. Prentice-Hall, 1999.

[7] F. Pentaris and Y. Ioannidis. Query Optimization in Distributed Networks of Autonomous Database Systems. *ACM Transactions on Database Systems (TODS)*, 31(2):537–583, 2006.

[8] F. Pentaris and Y. Ioannidis. Autonomic Query Allocation Based on Microeconomics Principles. In *Proceedings of the International Confeference on Data Engineering (ICDE)*, 2007.

[9] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. SQLB: A Query Allocation Framework for Autonomous Consumers and Providers. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, 2007.

[10] T. W. Sandholm. *Multiagent Systems, a modern approach to Distributed Artificial Intelligence*, chapter Distributed Rational Decision Making. The MIT Press, 2001.

[11] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of the Multimedia Computing and Networking Conference*, 2002.

[12] Y. Shoham and M. Tennenholtz. Fair Imposition. In *Proceedings of International Joint Conference on Aritficial Intelligence (IJCAI)*, 2001.

[13] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *Journal on Very Large Data Bases (VLDBJ)*, 5(1):48–63, 1996.

[14] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Finance*, 16(1), 1961.

[15] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2003.